

单元测验查看

第五章 二叉树后半部分 (5.5~5.7) 测验

1 下列关于二叉搜索树的说法正确的有

Which sentences of the followings are right about binary search tree:

(多选3 分)

- ☐ A. 二叉搜索树按照中序遍历将各结点打印出将各结点打印出来，将得到按照由小到大的排列。(正确答案)
If we print a binary search tree's nodes according its infix order, the sequence will be from small to large.

解析： 这是二叉搜索树的基本性质。
It is a basic characterization of a binary search tree
- ☐ B. 如果结点x的左子树有右子树，则存在某个结点的值介于结点x的值和x左儿子的值之间，并且这个结点在x的左子树之中。(正确答案)
If the left child tree of a node x has a right child tree, then there exists some node whose value is between the value of node x and the value of its left child node, and this node is on the left child tree of node x.

解析： 这样的结点就位于x的左子树的右子树中。This kind of nodes are on the right child tree of the left child tree of node x.
- ☐ C. 当根结点没有左儿子时，根结点一定是值最小的结点。(正确答案)
If the root node doesn't have left child, it must be the node with the smallest value.

解析： 右子树中的结点的值都大于根结点，所以根结点的值是最小的。
The values of nodes on the right child tree are all larger than the value of the root node, so the value of the root node has the smallest value.
- ☐ D. (错误答案)
二叉搜索树一定是满二叉树。A binary search tree must be a full binary tree.

解析： 不一定。如果对于一个结点存在值比它大的结点，但不存在比它小的，这时它可能就只有一个儿子。
- ☐ E. 二叉搜索树一定是完全二叉树。(错误答案)
A binary search tree must be a complete binary tree.

解析： 不一定。按照从小到大的顺序依次插入一些值（数量超过1个），就可以让二叉搜索树变成一条链，这样显然不是完全二叉树。
Not necessarily. If we insert some values (more than one value) from small to large, then the binary search tree will become a chain. Obviously it isn't a complete binary tree.
- ☐ F. 从根结点一直沿右儿子向下找不一定能找到树中值最大的结点。(错误答案)
Along the right child of nodes all the time from the root node, it is possible that we couldn't find out the node with the largest value.

解析： 右子树中的结点的值都大于根结点，所以根结点的值是最小的。
The values of nodes on the right child tree are all larger than the value of the root node, so the value of the root node has the smallest value.

2 如果按关键码值递增的顺序依次将n个关键码值插入到二叉搜索树中，如果对这样的二叉搜索树进行检索时，每次检索的字符都等概率的从这n个关键码值中选取，平均比较次数为多少？

If we insert n key values to a binary search tree successively from small to large, when we search this binary search tree, each time the search character is selected from these n key values with the same possibility, then how many times will the comparison be on average?

(填空2 分)

文字精确: (n+1)/2 或 (1+n)/2

3 从空二叉树开始，严格按照二叉搜索树的插入算法（不进行旋转平衡），逐个插入关键码{18,73,10,5,68,99,27,41,51,32,25}构造出一棵二叉搜索树，对该二叉搜索树按照前序遍历得到的序列为？（答案中每两个元素之间用一个空格隔开）

From a null binary tree, insert key values {18, 73, 10, 5, 68, 99, 27, 41, 51, 32, 25} successively according to the insertion algorithm of a binary search tree strictly (no rotation and balance) to construct a binary search tree. Please write down the sequence of preorder of this binary search tree. (There is one blank space between two elements)

(填空2 分)

文字精确: 18 10 5 73 68 27 25 41 32 51 99

4 从空二叉树开始，严格按照二叉搜索树的插入算法（不进行旋转平衡），逐个插入关键码{18,73,10,5,68,99,27,41,51,32,25}构造出一棵二叉搜索树，对该二叉搜索树按照后序遍历得到的序列为？（答案中每两个元素之间用一个空格隔开）

From a null binary tree, insert key values {18, 73, 10, 5, 68, 99, 27, 41, 51, 32, 25} successively according to the insertion algorithm of a binary search tree strictly (no rotation and balance) to construct a binary

在线客服

search tree. Please write down the sequence of post order of this binary search tree. (There is one blank space between two elements)

(填空2 分)

文字精确：5 10 25 32 51 41 27 68 99 73 18

5 从空二叉树开始，严格按照二叉搜索树的插入算法（不进行旋转平衡），逐个插入关键码构造出一棵二叉树，以怎样的顺序插入关键码集合{14, 32, 47, 6, 9, 12, 78, 63, 29, 81}可以使得树的深度最小？请依次写出插入到树中的元素，每两个元素之间用一个空格隔开。
如果有多组满足要求的方案，请使得你的答案中先插入的元素尽可能的小。

From a null binary tree, insert key values successively according to the insertion algorithm of a binary search tree strictly (no rotation and balance) to construct a binary search tree. What is the insertion sequence that could make the tree have a smallest depth with a key value set {14, 32, 47, 6, 9, 12, 78, 63, 29, 81}? Please write down the elements successively, and there is one blank space between two elements. If there are more than one answer that meet the condition, please make the element which needs to be inserted first as small as possible in your answer.

(填空2 分)

文字精确：12 6 9 47 29 14 32 78 63 81

解析：通过 $\log_2(10)=4$ 可以得到树的最小层数。然后因为需要保证先插入的元素尽可能的小，所以可以使得右子树尽可能的满。构造出这样一棵二叉搜索树后，按照前序遍历可以得出答案。 12 9 14 32 78 63 81

6 下列关于堆的说法正确的有:

Which sentences of the followings are right:

(多选3 分)

☐ A. 堆一定是完全二叉树。A heap must be a complete binary tree.(正确答案)

解析：定义如此。According to its defination

☐ B. 最小堆中，某个结点左子树中最大的结点可能比右子树中最小的结点小。In a minimum heap, the largest value on some node's left child tree could be possibly smaller than the smallest value of its right child tree. (正确答案)

解析：堆中一个结点的左儿子和右儿子并没有严格的大小关系，所以存在这种情况。
There is not a strict rule to stipulate which should be larger between the value of the left child and the right child of a node in a heap. So the condition exists.

☐ C. 使用筛选法建堆要比将元素一个一个插入堆来建堆效率高。Screening method has a higher efficiency than inserting elements one by one while constructing a heap. (正确答案)

解析：筛选法建堆的时间复杂度为 $O(n)$ ，而一个一个插入建堆时间复杂度为 $O(n\log n)$ 。其中， n 为堆中元素个数。
The complexity of screening method is $O(n)$, while inser

☐ D. 堆一定是满二叉树。A heap must be a full binary tree.(错误答案)

解析：有些堆的倒数第二层有部分结点只有一个儿子。
Some nodes on the second layer in inverted order in some heaps only have one child.

☐ E. 最小堆中，最下面一层最靠右的结点一定是权值最大的结点。In a minimum heap, the rightest node on the nethermost layer must be the node with the largest value. (错误答案)

解析：不一定。最小堆只保证了每个结点都比它的两个儿子都小，所以它左儿子的值可能比右儿子的大，所以最大的结点不一定靠右。
Not necessarily. A minimum heap could only guarantee than the value of each node is smaller than that of its two children, so the value of its left child is possibly larger than that of its right child

☐ F. 堆是实现优先队列的惟一方法。A heap is the only method to implement a priority queue.(错误答案)

解析：堆只是实现优先队列的一种方法。用普通的队列也可以实现优先队列，只是效率比较低。
A heap is one of the methods to implement a priority queue. A basic queue could also be used to implement a priority queue, with a relatively low efficiency.

7 对于关键码序列{38, 64, 52, 15, 73, 40, 48, 55, 26, 12}, 用筛选法建最小值堆，若一旦发现逆序对就进行交换，共需要交换元素多少次？

For the key value sequence {38, 64, 52, 15, 73, 40, 48, 55, 26, 12}, use the screening method to constuct a minimum heap, if we exchange them when we find reversed order, then how many times should we exchange them?

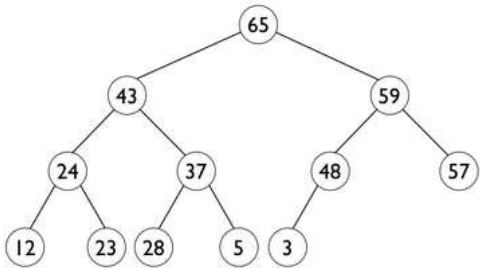
(填空2 分)

数值精确：6

解析：73和12进行交换，52和40进行交换，64和12进行交换，38和12进行交换，38和15进行交换，38和26进行交换，一共6次。Exchange 73 and 12, exchange 52 and 40, exc and 12, exchange 38 and 15, exchange 38 and 26, totally 6 times.

8 对于如下图所示的最大堆，删除掉最大的元素后，堆的前序遍历结果是

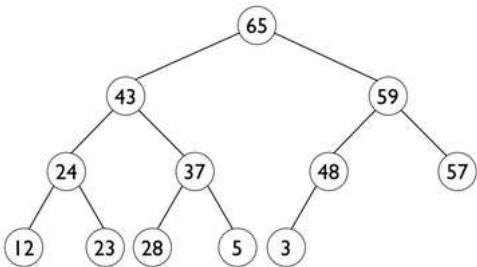
For the following maximum heap, after deleting the maximum element, the preorder traversal sequence is
请依次写出插入到树中的元素，每两个元素之间用一个空格隔开。
Please write down the elements successively, and there is one blank space between two elements.



(填空2 分)
文字精确: 59 43 24 12 23 37 28 5 57 48 3

9

对于如下图所示的最大堆，删除掉最大的元素后，堆的后序遍历结果是
For the following maximum heap, after deleting the maximum element, the post order traversal sequence is



(填空2 分)
文字精确: 12 23 24 28 5 37 43 48 3 57 59

10 下列关于Huffman树和Huffman编码的说法正确的有:

Which sentences of the followings are right about Huffman tree and Huffman code:

(多选3 分)

- ☐ A. Huffman树一定是满二叉树。A Huffman tree must be a full binary tree.(正确答案)
解析: 在建立的过程中，每次都选取两棵子树进行合并，所以所有结点要么有两个儿子，要么没有儿子。
While constructing the tree, we merge two child trees each time, so for all nodes, they have two children or don't have any child.
- ☐ B. Huffman编码是一种前缀编码。Huffman code is a kind of prefix code.(正确答案)
解析: Huffman树中，所有需要编码的内容都在叶结点中，所以任何内容的编码都不会是其它内容编码的前缀。
In a Huffman tree, all content which needs to be coded is on the leaf nodes, so codes of any content can't be prefix of codes of other content.
- ☐ C. 对于同样的一组权值两两不同的内容可以得到不同的Huffman编码方案。Different content with the same group of weights can get different Huffman codes. (正确答案)
解析: 把某一个结点往左子树编码0，往右子树编码1反过来就可以得到另外一组编码方案。Code a node's left child tree with 0 and its right child tree with 1. And vice versa, we get another group of codes.
- ☐ D. Huffman树一定是完全二叉树。A Huffman tree must be a complete binary tree.(错误答案)
解析: 取一棵是完全二叉树的Huffman树翻转过来，这棵树依然是Huffman树，但是已经不再是完全二叉树了。
We turn around a Huffman tree which is already a complete binary tree and it is also a Huffman tree, but it isn't a complete binary tree any more.
- ☐ E. Huffman编码中所有编码都是等长的。All codes in a Huffman code have the same length.(错误答案)
解析: Huffman树的叶结点并不一定在同一层，所以Huffman编码不等长。
Since leaf nodes in a Huffman tree could be on different layers, Huffman codes could have different lengths.
- ☐ F. 使用频率越高的字母，Huffman编码越长。The higher a letter's frequency is, the longer its Huffman code is. (错误答案)
解析: 频率越高，Huffman编码应该越短，这样才能提高编码效率。
The higher a letter's frequency is, the shorter its Huffman code is.

11

一组包含不同权的字母已经对应好Huffman编码，如果某一个字母对应编码001,下面说法正确的有

A group of letters with different weights has corresponded with Huffman codes, if a letter's corresponding code is 001, which sentences of the followings are right:

(多选3 分)

- ☐ A. (正确答案)
以001开头的编码不可能对应其他字母。A code beginning with 001 couldn't correspond with other letters.
解析： 通过001在Huffman树中已经到达叶结点，往下不会再有其他字母。
In the Huffman tree, it already arrives the leaf node through 001, so there is no letters along
- ☐ B. (正确答案)
以01开头和1开头的代码肯定对应某个字母。Codes beginning with 01 or 1 must correspondgding with some letters.
解析： 不然的话，不需要用001编码这一字母，可以更短。
If not, we don't need to use 001 to code a letter, it could be shorter.
- ☐ C. (正确答案)
建好的Huffman树至少包含4个叶结点。The Huffman tree contains at least 4 leaf nodes.
解析： 至少包含对应于001、000、01和1的4个叶结点。
It contains at least 4 leaf nodes corresponding with 001,000,01 and 1.
- ☐ D. (错误答案)
以000开头的代码不可能对应任何字母。Codes beginning with 000 couldn't correspond with any letter.
解析： 肯定要对应字母，不然的话001对应的字母可以缩短编码到00。
It must correspond with a letter. Otherwise, we can use 00 to code the letter instead of 001.
- ☐ E. (错误答案)
编码0和00可能对应于其他字母。Code 0 and 00 could corresponding with other letters.
解析： Huffman编码是前缀编码，任何一个字母的编码都不可能其它字母编码的前缀。Huffman code is a kind of prefix code, so the code of any letter couldn't be the prefix of codes of other lett

12 下表展示了在一段文本中每个字母出现的次数。

The frequencies that each letter appears in a paragraph is represented as follow.

a	12
e	8
i	15
o	4
u	9

对于这段文本使用Huffman编码相较使用等长编码能够节约多少比特的空间？ Comparing to use codes that have the same length, how many bits of space could be saved when we use Huffman code for the paragraph?

a	12
e	8
i	15
o	4
u	9

(填空2 分)

数值精确： 36
解析： 可以得到使用Huffman编码此文本一共需要2×12+3×8+2×15+3×4+2×9=108比特。而使用定长编码编码5个字母，每个字母需要3位，所以总共需要3×(12+8+15+4+9)=144比特。所以总共比特。 Draw out the corresponding Huffman tree. We can know that for the text it needs 2*12+3*8+2*15+3*4+2*9=108 bits it total when we use Huffman code. Using the codes that have the sa code the 5 letters, each letter needs 3 bits, so it needs 3*(12+8+15+4+9)=144 bits in total. Therefore, we save 144-108=36 bits totally.

13

对于给定的一组权 $W=\{1,4,9,16,25,36,49,64,81,100\}$ ，构造一棵具有最小带权外部路径长度的三叉树，写出这棵树的带权外部路径长度。

For a given group of weights $W=\{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$, please construct a ternary tree with a minimum weighted route length and write down this weighted route length.

(填空2 分)

数值精确： 705

14 请阅读下面一段代码

Please read the following code

C++:

```
while (!aStack.empty() || pointer) {?
    while (pointer != NULL) {
        // 1号访问点 //No. 1 visiting point
        element.pointer = pointer; element.tag = Left;
        aStack.push(element);
        pointer = pointer->leftchild();
    }
    element = aStack.top(); aStack.pop();
    pointer = element.pointer;
    if (element.tag == Left) {
        // 2号访问点 //No. 2 visiting point
        element.tag = Right;
        aStack.push(element);
        pointer = pointer->rightchild();
    } else {
        // 3号访问点 //No. 3 visiting point
        pointer = NULL;
    }
}
```

Python:

```
while not aStack.isEmpty() or pointer:
    while pointer != None:
        # 1号访问点 # No.1 visiting point
        element.pointer = pointer
        element.tag = Left
        aStack.push(element)
        pointer = pointer.leftchild
    element = aStack.pop()
    pointer = element.pointer
    if element.tag == Left:
        # 2号访问点 # No.2 visiting point
        element.tag = Right
        aStack.push(element)
        pointer = pointer.rightchild
    else:
        # 3号访问点 # No.3 visiting point
        pointer = None
```

若此段代码的作用是用来进行前序遍历，那么应该在几号访问点进行访问？（只需要填写数字）

if this code is used to do a preorder traversal, which visiting point should be visited? (You only need to write down the number)

(填空2分)

数值精确: 1

解析: 使用深搜算法进行前序遍历，每达到一个结点就应该进行访问。 Use the depth-first search algorithm to do preorder traversal, and visit while arriving a node.

15 请阅读下面一段代码

Please read the following code

C++:

```
while (!aStack.empty() || pointer) {?
    while (pointer != NULL) {
        // 1号访问点 //No. 1 visiting point
        element.pointer = pointer; element.tag = Left;
        aStack.push(element);
        pointer = pointer->leftchild();
    }
    element = aStack.top(); aStack.pop();
    pointer = element.pointer;
    if (element.tag == Left) {
        // 2号访问点 //No. 2 visiting point
        element.tag = Right;
        aStack.push(element);
        pointer = pointer->rightchild();
    } else {
        // 3号访问点 //No. 3 visiting point
        pointer = NULL;
    }
}
```

Python:

```

while not aStack.isEmpty() or pointer:
    while pointer != None:
        # 1号访问点 # No.1 visiting point
        element.pointer = pointer
        element.tag = Left
        aStack.push(element)
        pointer = pointer.leftchild
    element = aStack.pop()
    pointer = element.pointer
    if element.tag == Left:
        # 2号访问点 # No.2 visiting point
        element.tag = Right
        aStack.push(element)
        pointer = pointer.rightchild
    else:
        # 3号访问点 # No.3 visiting point
        pointer = None

```

若此段代码的作用是用来进行中序遍历，那么应该在几号访问点进行访问？（只需要填写数字）

if this code is used to do an infix order traversal, which visiting point should be visited? (You only need to write down the number)

（填空2分）

数值精确：2

解析：中序遍历应该在访问完一个结点的左子树后对该结点进行访问。 In an infix order traversal, we should visit the node after visiting its left child tree.

16 请阅读下面一段代码

Please read the following code

C++:

```

while (!aStack.empty() || pointer) {
    while (pointer != NULL) {
        // 1号访问点 //No. 1 visiting point
        element.pointer = pointer; element.tag = Left;
        aStack.push(element);
        pointer = pointer->leftchild();
    }
    element = aStack.top(); aStack.pop();
    pointer = element.pointer;
    if (element.tag == Left) {
        // 2号访问点 //No. 2 visiting point
        element.tag = Right;
        aStack.push(element);
        pointer = pointer->rightchild();
    } else {
        // 3号访问点 //No. 3 visiting point
        pointer = NULL;
    }
}

```

Python:

```

while not aStack.isEmpty() or pointer:
    while pointer != None:
        # 1号访问点 # No.1 visiting point
        element.pointer = pointer
        element.tag = Left
        aStack.push(element)
        pointer = pointer.leftchild
    element = aStack.pop()
    pointer = element.pointer
    if element.tag == Left:
        # 2号访问点 # No.2 visiting point
        element.tag = Right
        aStack.push(element)
        pointer = pointer.rightchild
    else:
        # 3号访问点 # No.3 visiting point
        pointer = None

```

若此段代码的作用是用来进行后序遍历，那么应该在几号访问点进行访问？（只需要填写数字）

if this code is used to do an post order traversal, which visiting point should be visited? (You only need to write down the number)

（填空2分）

数值精确：3

解析：后序遍历应该在一个结点的左子树和右子树都访问完后访问该结点。 In an post order traversal, we should visit the node's left and right child tree, then this node itself.